clq9761

Axis2创建WebService实例

博客分类：
- Java综合

WebServiceTomcatApacheWebXML

一、Axis2的下载和安装

1.可从http://ws.apache.org/axis2/ 下载Axis2的最新版本：
可以下载如下两个zip包：
axis2-1.5.4-bin.zip
axis2-1.5.4-war.zip
其中 axis2-1.5.4-bin.zip文件中包含了Axis2中所有的jar文件，
axis2-1.5.4-war.zip文件用于将WebService发布到Web容器中。

2.将axis2-1.5.4-war.zip文件解压到相应的目录，将目录中的axis2.war文件放到<Tomcat安装目录>\webapps目录中，
并启动Tomcat,在浏览器地址栏中输入如下的URL：
http://localhost:8080/axis2/，如看到axis2的主页面则安装成功。

二、编写和发布WebService

(1)用POJO形式发布（无需配置）

在Axis2中不需要进行任何的配置，就可以直接将一个简单的POJO发布成WebService。
其中POJO中所有的public方法将被发布成WebService方法。
示例代码如下：

Java代码

```
1.  public  class  HelloService  {
2.          public  String  sayHello(){
3.                  return  "hello";
4.          }
5.          public  String  sayHelloToPerson(String  name){
6.                  if(name==null){
7.                          name  =  "nobody";
8.                  }
9.                  return  "hello,"+name;
10.         }
11. }
```

编译HelloService类后，将HelloService.class文件放到<Tomcat安装目录>\webapps\axis2\WEB-INF\pojo目录中（如果没有pojo目录，则建立该目录）。现在我们已经成功将HelloService类发布成了WebService。
在浏览器地址栏中输入如下的URL：
http://localhost:8080/axis2/services/listServices

在浏览器地址栏中输入如下的两个URL来分别测试sayHelloToPerson和sayHello方法：
1.http://localhost:8080/axis2/services/HelloService/sayHello
2.http://localhost:8080/axis2/services/HelloService/sayHelloToPerson?name=bill

页面显示如下结果：

Xml代码

```xml
<ns:sayHelloToPersonResponse  xmlns:ns="http://ws.apache.org/axis2">
        <return>hello,bill</return>
</ns:sayHelloToPersonResponse>
```

在编写、发布和测试WebService时应注意如下几点：
1. POJO类不能使用package关键字声明包。

2. Axis2在默认情况下可以热发布WebService，也就是说，将WebService的.class文件复制到pojo目录中时，Tomcat不需要重新启动就可以自动发布WebService。
如果想取消Axis2的热发布功能，可以打开<Tomcat安装目录>\webapps\axis2\WEB-INF\conf\axis2.xml，找到如下的配置代码：

Xml代码

```xml
<parameter  name="hotdeployment">true</parameter>
```

将true改为false即可。要注意的是，Axis2在默认情况下虽然是热发布，但并不是热更新.
也就是说，一旦成功发布了WebService，再想更新该WebService，就必须重启Tomcat。
这对于开发人员调试WebService非常不方便，因此，在开发WebService时，可以将Axis2设为热更新。
在axis2.xml文件中找到

Xml代码

```xml
<parameter  name="hotupdate">false</parameter>
```

将false改为true即可。

3. 在浏览器中测试WebService时，如果WebService方法有参数，需要使用URL的请求参数来指定该WebService方法
参数的值，请求参数名与方法参数名要一致，例如，要测试sayHelloToPerson方法，请求参数名应为name，如上面的URL所示。

4. 发布WebService的pojo目录只是默认的，如果读者想在其他的目录发布WebService，
可以打开axis2.xml文件，并在<axisconfig>元素中添加如下的子元素：

Xml代码

```xml
<deployer  extension=".class"  directory="my"  class="org.apache.axis2.deployment.POJODeployer"/>
```

上面的配置允许在<Tomcat安装目录>\webapps\axis2\WEB-INF\my目录中发布WebService。
例如，将本例中的HelloService.class复制到my目录中也可以成功发布
（但要删除pojo目录中的SimpleService.class，否则WebService会重名）。

(2)使用services.xml配置文件发布

用Axis2实现Web Service，虽然可以将POJO类放在axis2\WEB-INF\pojo目录中直接发布成Web Service，
这样做不需要进行任何配置，但这些POJO类不能在任何包中。这似乎有些不方便.
为此，Axis2也允许将带包的POJO类发布成Web Service。先实现一个POJO类，代码如下：

Java代码

```java
package com.sinosoft.webservice;
public class HelloServiceNew {
        public String sayHelloNew(){
                return "hello";
        }
        public String sayHelloToPersonNew(String name){
                if(name==null){
                        name = "nobody";
                }
                return "hello,"+name;
        }
        public void updateData(String data){
                System.out.println(data+" 已更新。");
        }
}
```

要想将HelloServiceNew类发布成Web Service，需要一个services.xml文件，
这个文件需要放在META-INF目录中，该文件的内容如下：

Xml代码

```xml
<?xml version="1.0" encoding="UTF-8"?>
<service name="HelloServiceNew">
        <description>
                Web Service例子
        </description>
        <parameter name="ServiceClass">
                com.sinosoft.webservice.HelloServiceNew
        </parameter>
        <messageReceivers>
                <messageReceiver mep="http://www.w3.org/2004/08/wsdl/in-out"
                        class="org.apache.axis2.rpc.receivers.RPCMessageReceiver" />
                <messageReceiver mep="http://www.w3.org/2004/08/wsdl/in-only"
                        class="org.apache.axis2.rpc.receivers.RPCInOnlyMessageReceiver" />
        </messageReceivers>
</service>
```

其中<service>元素用于发布Web Service，一个<service>元素只能发布一个WebService类，
name属性表示WebService名，如下面的URL可以获得这个WebService的WSDL内容：
http://localhost:8080/axis2/services/HelloServiceNew?wsdl
其中name属性名就是上面URL中"?"和"/"之间的部分。
<description>元素表示当前Web Service的描述，<parameter>元素用于设置WebService的参数，
在这里用于设置WebService对应的类名。
在这里最值得注意的是<messageReceivers>元素，该元素用于设置处理WebService方法的处理器。
例如，sayHelloNew方法有一个返回值，因此，需要使用可处理输入输出的RPCMessageReceiver类，
而updateData方法没有返回值，因此，需要使用只能处理输入的RPCInOnlyMessageReceiver类。

使用这种方式发布WebService，必须打包成.aar文件，.aar文件实际上就是改变了扩展名的.jar文件。
现在建立了两个文件：HelloServiceNew.java和services.xml。
将HelloServiceNew.java编译，生成HelloServiceNew.class。
services.xml和HelloServiceNew.class文件的位置如下：
D:\ws\ com\sinosoft\webservice\HelloServiceNew.class
D:\ws\META-INF\services.xml
在windows控制台中进入ws目录，并输入如下的命令生成.aar文件.

```
jar cvf ws.aar .
```

实际上，.jar文件也可以发布webservice，但axis2官方文档中建议使用.aar文件发布webservice.
最后将ws.aar文件复制到<Tomcat安装目录>\webapps\axis2\WEB-INF\services目录中，
启动Tomcat后，就可以调用这个WebService了。

另外services.xml文件中也可以直接指定WebService类的方法，如可以用下面的配置代码来发布WebService

Xml代码

```xml
<service name=" HelloServiceNew ">
<description>
        Web Service例子
</description>
<parameter name="ServiceClass">
        com.sinosoft.webservice.HelloServiceNew
</parameter>
<operation name="sayHello">
        <messageReceiver class="org.apache.axis2.rpc.receivers.RPCMessageReceiver"/>
</operation>
<operation name="updateData">
        <messageReceiver
                class="org.apache.axis2.rpc.receivers.RPCInOnlyMessageReceiver"/>
        </operation>
</service>
```

如果想发布多个WebService，可以使用<serviceGroup>元素

Xml代码

```xml
<serviceGroup>
<service name="myService1">
        ...
</service>
<service name="myService2">
        ...
</service>
</serviceGroup>
```

中间省略的代码同上面services.xml文件的配置。

三、 用Java实现调用WebService的客户端程序

WebService是为程序服务的，只在浏览器中访问WebService是没有意义的。调用WebService的客户端代码如下：

Java代码

```java
import javax.xml.namespace.QName;
import org.apache.axis2.AxisFault;
import org.apache.axis2.addressing.EndpointReference;
import org.apache.axis2.client.Options;
import org.apache.axis2.rpc.client.RPCServiceClient;
public class TestMain {
public static void main(String args[]) throws AxisFault{
      //    使用RPC方式调用WebService
        RPCServiceClient serviceClient = new RPCServiceClient();
        Options options = serviceClient.getOptions();
        //    指定调用WebService的URL
        EndpointReference targetEPR = new EndpointReference(
                        "http://localhost:8080/axis2/services/HelloService");
        options.setTo(targetEPR);
        //    指定sayHelloToPerson方法的参数值
        Object[] opAddEntryArgs = new Object[] {"美女"};
        //    指定sayHelloToPerson方法返回值的数据类型的Class对象
        Class[] classes = new Class[] {String.class};
        //    指定要调用的sayHelloToPerson方法及WSDL文件的命名空间
        QName opAddEntry = new QName("http://ws.apache.org/axis2", "sayHelloToPerson");
        //    调用sayHelloToPerson方法并输出该方法的返回值
        System.out.println(serviceClient.invokeBlocking(opAddEntry, opAddEntryArgs, classes)
```

```
    [0]);
23. }
24. }
```

输出结果为：
hello,美女

在编写客户端代码时应注意如下几点：

1. 客户端代码需要引用很多Axis2的jar包，如果读者不太清楚要引用哪个jar包，
   可以在Eclipse的工程中引用Axis2发行包的lib目录中的所有jar包。

2. 在本例中使用了RPCServiceClient类的invokeBlocking方法调用了WebService中的方法。
   invokeBlocking方法有三个参数，其中第一个参数的类型是QName对象，表示要调用的方法名；
   第二个参数表示要调用的WebService方法的参数值，参数类型为Object[]；
   第三个参数表示WebService方法的返回值类型的Class对象，参数类型为Class[]。
   当方法没有参数时，invokeBlocking方法的第二个参数值不能是null，而要使用new Object[]{}。

3. 如果被调用的WebService方法没有返回值，应使用RPCServiceClient类的invokeRobust方法，
   该方法只有两个参数，它们的含义与invokeBlocking方法的前两个参数的含义相同。

4. 在创建QName对象时，QName类的构造方法的第一个参数表示WSDL文件的命名空间名，
   也就是<wsdl:definitions>元素的targetNamespace属性值。

四、用wsdl2java简化客户端的编写

Axis2提供了一个wsdl2java.bat命令可以根据WSDL文件自动产生调用WebService的代码。
wsdl2java.bat命令可以在<Axis2安装目录>/bin目录中找到。
在使用wsdl2java.bat命令之前需要设置AXIS2_HOME环境变量，该变量值是<Axis2安装目录>。
在Windows控制台输出如下的命令行来生成调用WebService的代码：

```
%AXIS2_HOME%\bin\wsdl2java -uri http://localhost:8080/axis2/services/HelloService?wsdl
          -p client -s -o stub
```

其中-url参数指定了wsdl文件的路径，可以是本地路径，也可以是网络路径。
-p参数指定了生成的Java类的包名，-o参数指定了生成的一系列文件保存的根目录。
在执行完上面的命令后，就会发现在当前目录下多了个stub目录，
在stub/src/client目录可以找到一个HelloServiceStub.java文件，
该文件复杂调用WebService，可以在程序中直接使用这个类，代码如下：

Java代码

```
1.  package  client;
2.  public  class  StupTest  {
3.          public  static  void  main(String[]  args)  throws  Exception
4.          {
5.                  HelloServiceStub  stub  =  new  HelloServiceStub();
6.                  HelloServiceStub.SayHelloToPerson  gg  =  new  HelloServiceStub.SayHelloToPerson();
7.                  gg.setName("美女");
8.                  System.out.println(  stub.sayHello().get_return());
9.                  System.out.println(stub.sayHelloToPerson(gg).get_return());
10.         }
11. }
```

输出结果如下：
hello
hello,美女

上面的代码大大简化了调用WebService的步骤，并使代码更加简洁。
但要注意的是，wsdl2java.bat命令生成的Stub类将WebService方法的参数都封装在了相应的类中，
类名为方法名，例如，sayHelloToPerson方法的参数都封装在了SayHelloToPerson类中，
要想调用sayHelloToPerson方法，必须先创建SayHelloToPerson类的对象实例。